

Resolução lista 3

Grupo: Eduardo Garcez, Davi Augusto, Gabriel Netto, João Arend e Luan Frederico.

Contents

Exercício 4	1
a	2
b	3
c	5
d	7
Exercício 5	12
a	12
b	14
c	14
d	17
e	19

Exercício 4

As doze observações a seguir são de um estudo de confiabilidade de componentes industriais: 0.56, 2.26, 1.90, 0.94, 1.40, 1.39, 1.00, 1.45, 2.32, 2.08, 0.89, 1.68.

Um modelo Weibull com parâmetros α e η é considerado apropriado:

$$f(x|\alpha, \eta) = \alpha\eta x^{\alpha-1}e^{-\eta x^\alpha}, \quad x, \alpha, \eta > 0$$

Considere a distribuição a priori, com valores especificados $\beta = 0.01$ e $\xi = 1$, sendo

$$g(\alpha, \eta) \propto e^{-\alpha}\eta^{\beta-1}e^{-\xi\eta}$$

A distribuição a posteriori de (α, η) dado os dados tem densidade

$$g(\alpha, \eta|x) \propto (\alpha\eta)^n \left(\prod_{i=1}^n x_i \right)^{\alpha-1} \exp[-\eta \sum_{i=1}^n x_i^\alpha] \times g(\alpha, \eta)$$

Para obter uma amostra da densidade a posteriori, pode-se usar o algoritmo MetropolisHastings (M-H) com densidade proposta

$$q(\alpha^c, \eta^c | \alpha, \eta) = \frac{1}{\alpha\eta} \exp\left[-\frac{\alpha^c}{\alpha} - \frac{\eta^c}{\eta}\right]$$

que é um produto de duas distribuições exponenciais independentes com médias α e η .

a

Calcule a probabilidade de aceitação no t-ésimo passo da cadeia M-H e explique como a cadeia deve ser gerada.

1. Probabilidade de aceitação no passo t

Estando em um estado atual $\theta = (\alpha, \eta)$, propomos um novo estado candidato $\theta^c = (\alpha^c, \eta^c)$.

A probabilidade de aceitação é dada por $\rho = \min(1, R)$. Sendo R a **Razão de Metropolis-Hastings**.

$$R = \underbrace{\frac{g(\theta^c|x)}{g(\theta|x)}}_{\text{Razão das Posteriori}} \times \underbrace{\frac{q(\theta|\theta^c)}{q(\theta^c|\theta)}}_{\text{Razão das Propostas}}$$

nota: $g(\theta^c|x) = g(\alpha^c, \eta^c|x)$ é a densidade da posteriori.

- **Razão das Posteriori:** Compara o quão “bom” (provável) é o novo ponto candidato em relação ao ponto atual, de acordo com a nossa distribuição alvo (a posteriori).
- **Razão das Propostas:** Corrige qualquer assimetria na nossa forma de propor. Se for mais fácil pular de θ para θ^c do que o contrário, este termo ajusta a probabilidade para garantir que não fiquemos “presos” em uma região só porque é fácil chegar lá.

1.1 Aplicação no problema

Posteriori: $g(\alpha, \eta|x) \propto (\alpha\eta)^n (\prod x_i)^{\alpha-1} \cdot \exp[-\eta \sum x_i^\alpha] \cdot g(\alpha, \eta)$

Priori: $g(\alpha, \eta) \propto e^{-\alpha\beta - \xi\eta}$

Proposta: $q(\theta^c|\theta) = q(\alpha^c, \eta^c|\alpha, \eta) = \frac{1}{\alpha\eta} \exp[-\frac{\alpha^c}{\alpha} - \frac{\eta^c}{\eta}]$

Constantes: $n = 12$, $\beta = 0.01$, $\xi = 1$.

•

1.1.1 Razão das posteriori

Denominamos $P = \prod_{i=1}^n x_i$ e $S(\alpha) = \sum_{i=1}^n x_i^\alpha$.

$$R_g = \frac{g(\theta^c|x)}{g(\theta|x)} = \frac{(\alpha^c)^n (\eta^c)^n P^{\alpha^c-1} e^{-\alpha^c\beta} e^{-\eta^c(S(\alpha^c)+\xi)}}{(\alpha)^n (\eta)^n P^{\alpha-1} e^{-\alpha\beta} e^{-\eta(S(\alpha)+\xi)}}$$

Agrupando os termos, obtemos:

$$R_g = \frac{g(\theta^c|x)}{g(\theta|x)} = \left(\frac{\alpha^c}{\alpha} \right)^n \left(\frac{\eta^c}{\eta} \right)^n \left(P^{\alpha^c-\alpha} \right) \left(e^{-\beta(\alpha^c-\alpha)} \right) \left(\frac{e^{-\eta^c(S(\alpha^c)+\xi)}}{e^{-\eta(S(\alpha)+\xi)}} \right)$$

1.1.2 Razão das propostas

$$R_q = \frac{q(\theta|\theta^c)}{q(\theta^c|\theta)} = \frac{\frac{1}{\alpha^c\eta^c} \exp[-\frac{\alpha}{\alpha^c} - \frac{\eta}{\eta^c}]}{\frac{1}{\alpha\eta} \exp[-\frac{\alpha^c}{\alpha} - \frac{\eta^c}{\eta}]}$$

Rearranjando os termos:

$$R_q = \frac{q(\theta|\theta^c)}{q(\theta^c|\theta)} = \left(\frac{\alpha\eta}{\alpha^c\eta^c} \right) \cdot \exp \left(\left(\frac{\alpha^c}{\alpha} + \frac{\eta^c}{\eta} \right) - \left(\frac{\alpha}{\alpha^c} + \frac{\eta}{\eta^c} \right) \right)$$

E a **probabilidade de aceitação final no t-ésimo** passo é:

$$\rho_{acc} = \min(1, R) = \min(1, R_g \times R_q)$$

2. Como a cadeia deve ser gerada

1. Início:

Escolha valores iniciais (chutes) para os parâmetros, $\alpha^{(0)}$ e $\eta^{(0)}$. Estes devem ser valores positivos, pois o modelo Weibull exige $\alpha, \eta > 0$.

Defina o número total de iterações, N .

2. Iteração:

2.1 A partir do estado atual $\theta^{(t)}$, amostre o candidato independente em cada componente:

- $\alpha^{(c)} \sim \text{Exp}(\text{média} = \alpha^{(t)})$;
- $\eta^{(c)} \sim \text{Exp}(\text{média} = \eta^{(t)})$

2.2 Calcular a probabilidade de aceitação:

Calcule R (ou $\text{Log}(R)$) usando a fórmula que derivamos acima.

Determine a probabilidade de aceitação: $\rho_{acc} = \min(1, R)$.

2.3 Aceite ou rejeite:

Gere um número aleatório u de uma distribuição Uniforme(0, 1)

- Se $u < \rho_{acc}$: aceitamos, $\theta^{(t+1)} = \theta^{(c)}$;
- C.C. rejeitamos, $\theta^{(t+1)} = \theta^{(t)}$

3. Final

Após N iterações, a sequência $\{\theta^{(1)}, \dots, \theta^{(N)}\}$ é cadeia.

b

Construa um código comentado para gerar uma amostra da distribuição a posteriori a partir do algoritmo obtido no item anterior.

```
# Dados do estudo de confiabilidade
x <- c(0.56, 2.26, 1.90, 0.94, 1.40, 1.39, 1.00, 1.45, 2.32, 2.08, 0.89, 1.68)
n <- length(x)

# Hiperparâmetros da priori g(alpha, eta)
beta <- 0.01
xi <- 1

# Pré-calcular valores constantes dos dados
log_prod_x <- sum(log(x))

# --- FUNÇÃO LOG-POSTERIORI ---
# Esta função calcula o logaritmo da densidade a posteriori g(alpha, eta | x)
# log(g(alpha, eta | x)) = constante + log(g(alpha, eta | x)_núcleo)
#
# O núcleo da posteriori que derivamos em 4(a) foi:
# g(...) proporcional [alpha^n * (prod(x_i))^(alpha-1) * exp(-alpha)] * # [eta^(n+beta-1) * exp(-eta *
# #
# O logaritmo disso é:
# log(g(...)) = n*log(alpha) + (alpha-1)*log(prod(x_i)) - alpha +
#               (n+beta-1)*log(eta) - eta * (sum(x_i^alpha) + xi)
```

```

#-----#
log_posterior <- function(alpha, eta, x, n, beta, xi, log_prod_x) {

  # Garantir que os parâmetros sejam positivos (requisito do modelo)
  if (alpha <= 0 || eta <= 0) {
    return(-Inf) # Retorna -Infinito no log, que é prob. zero
  }

  # Calcular sum(x_i^alpha)
  sum_x_alpha <- sum(x^alpha)

  # Parte do log-posterior relacionada a alpha
  log_alpha_part <- n * log(alpha) + (alpha - 1) * log_prod_x - alpha

  # Parte do log-posterior relacionada a eta
  log_eta_part <- (n + beta - 1) * log(eta) - eta * (sum_x_alpha + xi)

  # Retorna a soma (log(A*B) = log(A) + log(B))
  return(log_alpha_part + log_eta_part)
}

# --- ALGORITMO METROPOLIS-HASTINGS ---
amostrador_mh <- function(N, alpha_start, eta_start, x, n, beta, xi, log_prod_x) {

  # 1. Inicialização
  # Criar vetores para armazenar a cadeia de Markov
  alpha_chain <- numeric(N)
  eta_chain   <- numeric(N)

  # Definir o estado inicial (t=1)
  alpha_chain[1] <- alpha_start
  eta_chain[1]   <- eta_start

  # Contador para a taxa de aceitação
  acceptance_count <- 0

  # 2. Loop de Iteração (do passo t=2 até N)
  for (t in 2:N) {

    # Obter o estado atual (t-1)
    alpha_t <- alpha_chain[t-1]
    eta_t   <- eta_chain[t-1]

    # 3. Proposta (Gerar o candidato)
    # Proposta: q(c/t) é Exp(media=alpha_t) * Exp(media=eta_t)
    # Em R, rexp(n, rate) usa taxa = 1/media
    alpha_c <- rexp(1, rate = 1 / alpha_t) # Candidato alpha^c
    eta_c   <- rexp(1, rate = 1 / eta_t)   # Candidato eta^c

    # 4. Cálculo da Razão de Aceitação (em log-escala)
    # log(R) = log(g(c/x)) - log(g(t/x)) + log(q(t/c)) - log(q(c/t))
  }
}

```

```

# Razão das Posteriores: log(g(c/x)) - log(g(t/x))
log_R_g <- log_posterior(alpha_c, eta_c, x, n, beta, xi, log_prod_x) -
  log_posterior(alpha_t, eta_t, x, n, beta, xi, log_prod_x)

# Razão das Propostas: log(q(t/c)) - log(q(c/t))
# q(t/c) = Exp(alpha_t / media=alpha_c) * Exp(eta_t / media=eta_c)
# q(c/t) = Exp(alpha_c / media=alpha_t) * Exp(eta_c / media=eta_t)
# Usamos dexp(x, rate, log = TRUE) que é log(rate) - rate*x

log_q_t_given_c <- dexp(alpha_t, rate = 1 / alpha_c, log = TRUE) +
  dexp(eta_t, rate = 1 / eta_c, log = TRUE)

log_q_c_given_t <- dexp(alpha_c, rate = 1 / alpha_t, log = TRUE) +
  dexp(eta_c, rate = 1 / eta_t, log = TRUE)

log_R_q <- log_q_t_given_c - log_q_c_given_t

# Razão total
log_R <- log_R_g + log_R_q

# 5. Decisão (Aceitar ou Rejeitar)
# A = min(1, R), então log(A) = min(0, log(R))
# Geramos log(u) de uma Uniforme(0,1), que é -Exp(1)

log_u <- log(runif(1, 0, 1))

if (log_u < log_R) {
  # Aceitar o candidato
  alpha_chain[t] <- alpha_c
  eta_chain[t] <- eta_c
  acceptance_count <- acceptance_count + 1
} else {
  # Rejeitar o candidato e repetir o estado anterior
  alpha_chain[t] <- alpha_t
  eta_chain[t] <- eta_t
}

# Imprimir a taxa de aceitação (para diagnóstico)
cat("Taxa de Aceitação:", acceptance_count / N, "\n")

# Retornar a cadeia completa
return(list(alpha = alpha_chain, eta = eta_chain))
}

```

c

Gere uma trajetória da cadeia a partir do código e dos dados, especificando o tamanho da cadeia, o período de burn-in, o thinning (lag) e os valores iniciais.

```

set.seed(52)

# Parâmetros especificados para a geração da cadeia
N_total      <- 25000 # Tamanho total da cadeia

```

```

burn_in      <- 5000 # Período de burn-in (amostras a descartar)
thinning     <- 20
alpha_inicial <- 1.0
eta_inicial   <- 1.0

cadeia_mh_bruta <- amostrador_mh(
  N           = N_total,
  alpha_start = alpha_inicial,
  eta_start   = eta_inicial,
  x            = x,
  n            = n,
  beta         = beta,
  xi           = xi,
  log_prod_x   = log_prod_x
)

# --- PROCESSAMENTO DA CADEIA (APLICANDO BURN-IN E THINNING) ---
# Primeiro, removemos o período de burn-in
indices_pos_burnin <- (burn_in + 1):N_total

alpha_pos_burnin <- cadeia_mh_bruta$alpha[indices_pos_burnin]
eta_pos_burnin   <- cadeia_mh_bruta$eta[indices_pos_burnin]

# Segundo, aplicamos o thinning
# Criamos uma sequência de índices para "pular" de 'thinning' em 'thinning'
indices_thinning <- seq(from = 1,
                         to = length(alpha_pos_burnin),
                         by = thinning)

# A nossa trajetória final para inferência é:
alpha_final <- alpha_pos_burnin[indices_thinning]
eta_final   <- eta_pos_burnin[indices_thinning]

# O número final de amostras
N_final <- length(alpha_final)

# RESULTADO: TRAJETÓRIA GERADA
cat("\n--- Especificações da Geração da Cadeia ---\n")
cat("Tamanho total (N_total):", N_total, "\n")
cat("Valores Iniciais: (alpha_0 =", alpha_inicial, ", eta_0 =", eta_inicial, ") \n")
cat("Período de Burn-in:", burn_in, "amostras descartadas\n")
cat("Thinning (lag):", thinning, "(mantendo 1 a cada", thinning, "amostras)\n")
cat("Tamanho da amostra final (para inferência):", N_final, "amostras\n")
# A trajetória gerada está armazenada nas variáveis 'alpha_final' e 'eta_final'.
# Vamos ver as primeiras 6 amostras dessa trajetória:
cat("\nPrimeiras 5 amostras da trajetória final:\n")
print(head(data.frame(alpha = alpha_final, eta = eta_final), 5))
cat("\nÚltimas 5 amostras da trajetória final:\n")
print(tail(data.frame(alpha = alpha_final, eta = eta_final), 5))

## Taxa de Aceitação: 0.08352
##
## --- Especificações da Geração da Cadeia ---
## Tamanho total (N_total): 25000

```

```

## Valores Iniciais: (alpha_0 = 1 , eta_0 = 1 )
## Período de Burn-in: 5000 amostras descartadas
## Thinning (lag): 20 (mantendo 1 a cada 20 amostras)
## Tamanho da amostra final (para inferência): 1000 amostras
##
## Primeiras 5 amostras da trajetória final:
##      alpha      eta
## 1 2.816070 0.2457242
## 2 3.625726 0.1999837
## 3 2.148016 0.2842414
## 4 2.148016 0.2842414
## 5 2.148016 0.2842414
##
## Últimas 5 amostras da trajetória final:
##      alpha      eta
## 996 2.296040 0.3722157
## 997 2.296040 0.3722157
## 998 2.540923 0.2522317
## 999 2.540923 0.2522317
## 1000 2.540923 0.2522317

```

d

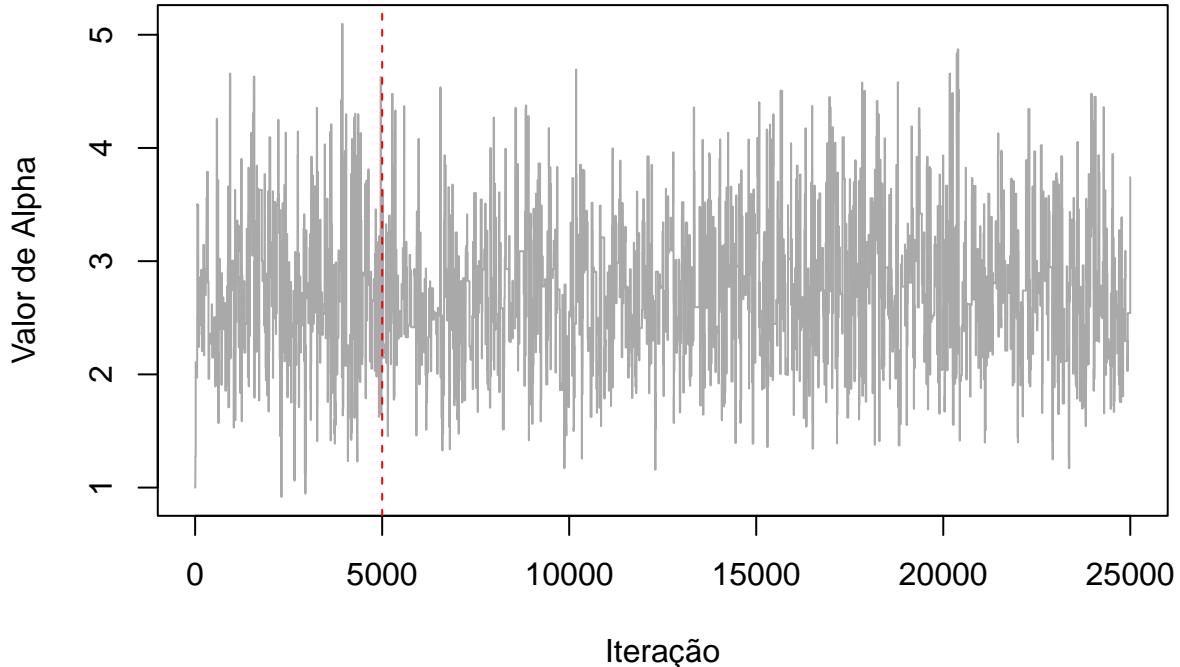
Avalie a convergência da cadeia gerada e realize a inferência para os parâmetros; utilize estimativas pontuais, intervalos de credibilidade e construa gráficos.

```

# 1. GRÁFICOS DE TRAJETÓRIA (TRACE PLOTS)
# Usamos a cadeia "bruta" ANTES do burn-in para ver o aquecimento
plot(cadeia_mh_bruta$alpha, type = 'l', col = "darkgray",
      main = "Trajetória de Alpha (Bruta)",
      xlab = "Iteração", ylab = "Valor de Alpha")
# Adiciona uma linha mostrando onde o burn-in termina
abline(v = burn_in, col = "red", lty = 2)

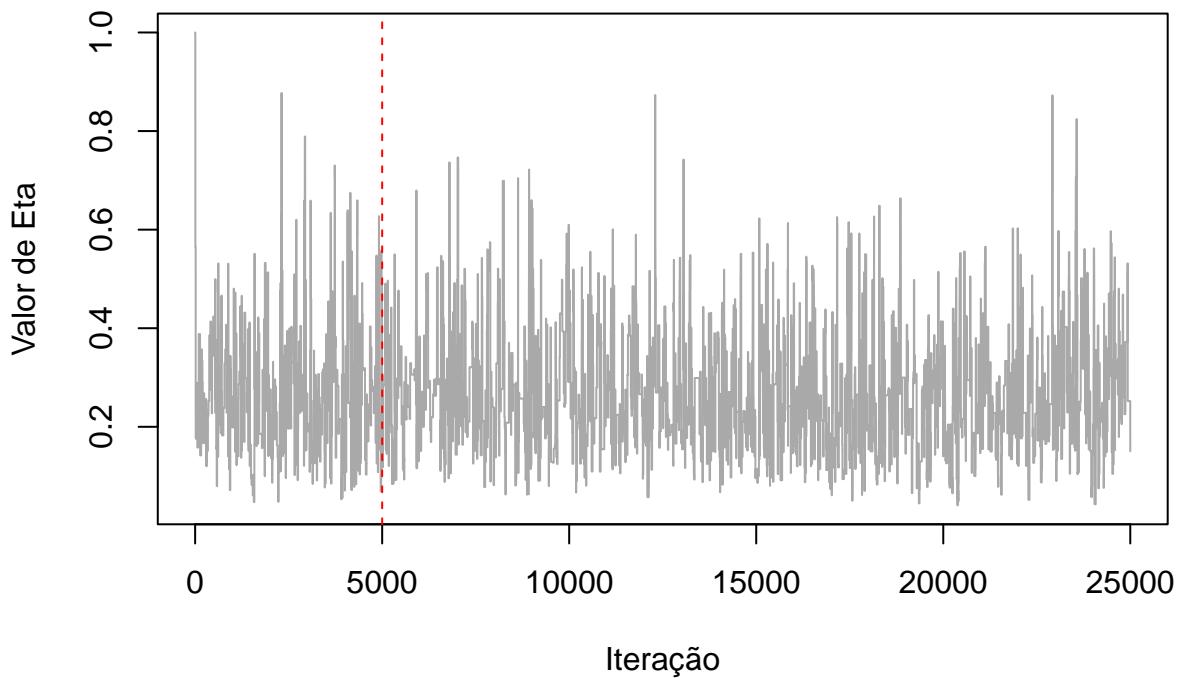
```

Trajetória de Alpha (Bruta)



```
plot(cadeia_mh_bruta$eta, type = 'l', col = "darkgray",
      main = "Trajetória de Eta (Bruta)",
      xlab = "Iteração", ylab = "Valor de Eta")
abline(v = burn_in, col = "red", lty = 2)
```

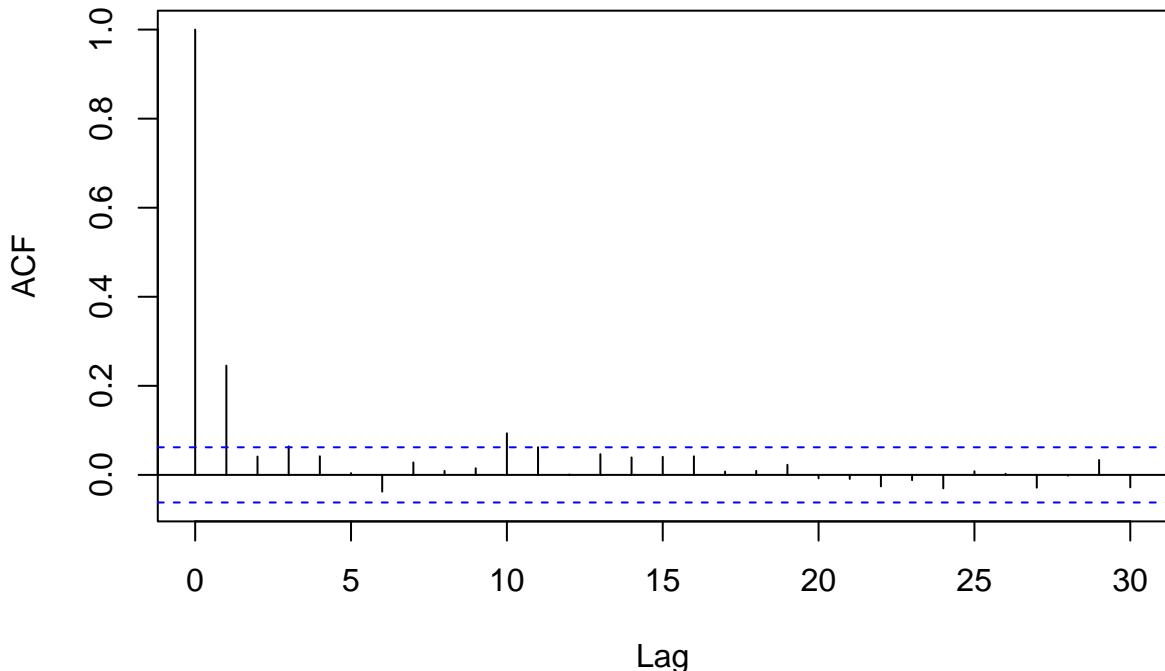
Trajetória de Eta (Bruta)



Idealmente, o gráfico deve parecer “estacionário” (sem tendências) após a linha vermelha do burn-in.

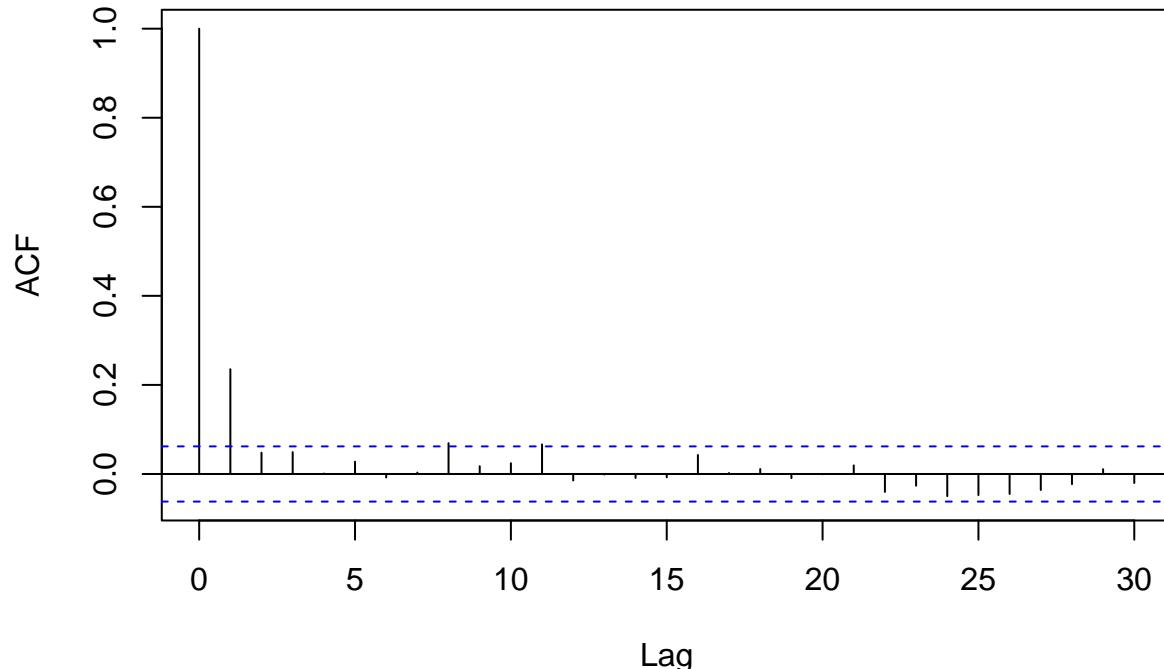
```
# 2. GRÁFICOS DE AUTOCORRELAÇÃO (ACF PLOTS)
# Usamos a cadeia "final" (com thinning) para verificar a independência
acf(alpha_final, main = "ACF de Alpha (Pós-Thinning)")
```

ACF de Alpha (Pós-Thinning)



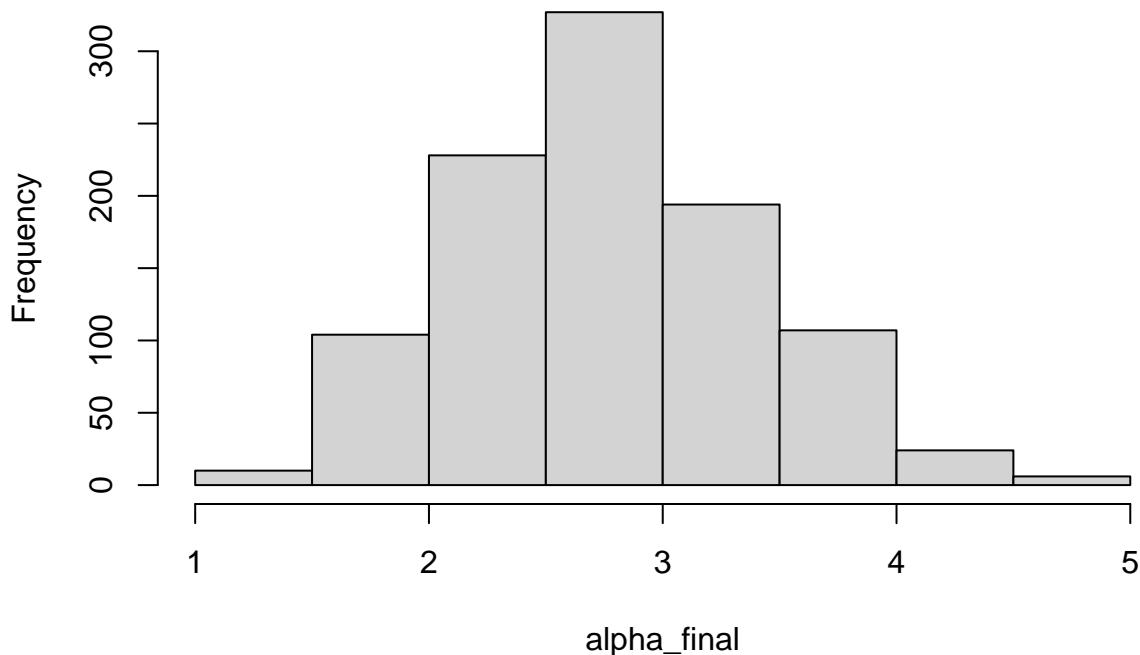
```
acf(eta_final, main = "ACF de Eta (Pós-Thinning)")
```

ACF de Eta (Pós-Thinning)



```
# Para Alpha  
hist(alpha_final)
```

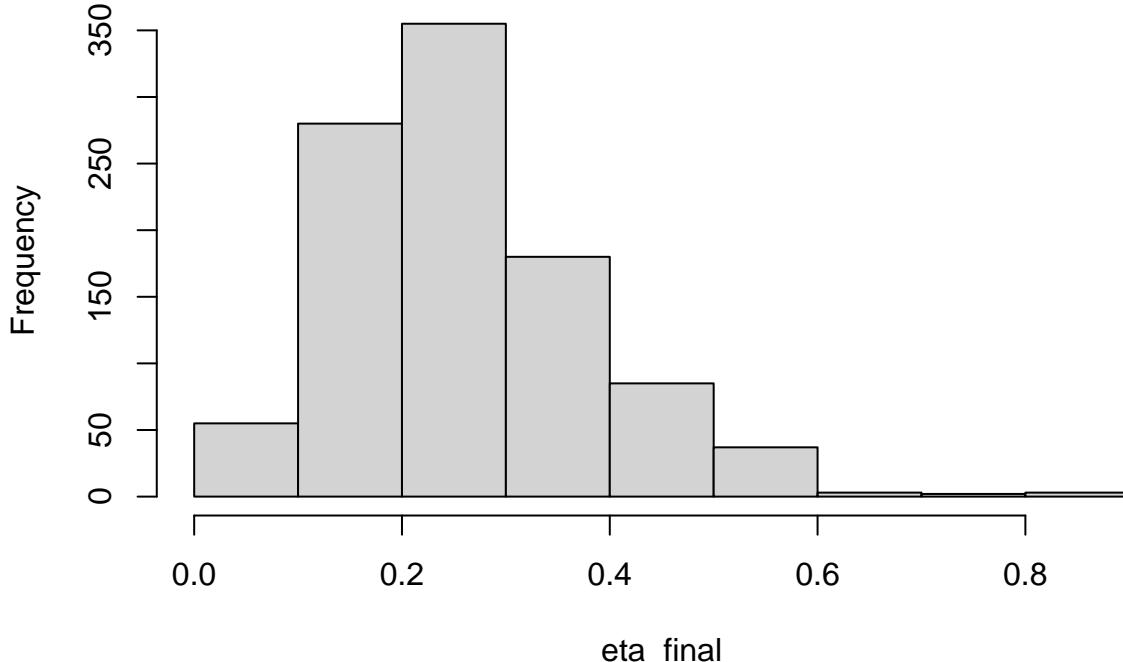
Histogram of alpha_final



```
media_alpha <- mean(alpha_final)  
mediana_alpha <- median(alpha_final)
```

```
# Para Eta
hist(eta_final)
```

Histogram of eta_final



```
media_eta <- mean(eta_final)
mediana_eta <- median(eta_final)

# INTERVALOS DE CREDIBILIDADE (95%)
# Usamos os quantis 2.5% e 97.5% da amostra final

ic_alpha <- quantile(alpha_final, probs = c(0.025, 0.975))
ic_eta <- quantile(eta_final, probs = c(0.025, 0.975))

# RESUMO DOS RESULTADOS

cat("\n--- Resumo da Inferência (Baseado em", N_final, "amostras) ---\n\n")

cat("Parâmetro: Alpha\n")
cat(" Média da Posterior:", round(media_alpha, 4), "\n")
cat(" Mediana da Posterior:", round(mediana_alpha, 4), "\n")
cat(" Intervalo de Credibilidade 95%:",
    "[", round(ic_alpha[1], 4), ", ", round(ic_alpha[2], 4), "]\n\n")

cat("Parâmetro: Eta\n")
cat(" Média da Posterior:", round(media_eta, 4), "\n")
cat(" Mediana da Posterior:", round(mediana_eta, 4), "\n")
cat(" Intervalo de Credibilidade 95%:",
    "[", round(ic_eta[1], 4), ", ", round(ic_eta[2], 4), "]\n\n")

##
```

```

## --- Resumo da Inferência (Baseado em 1000 amostras) ---
##
## Parâmetro: Alpha
## Média da Posterior: 2.7701
## Mediana da Posterior: 2.74
## Intervalo de Credibilidade 95%: [ 1.6441 , 4.0944 ]
##
## Parâmetro: Eta
## Média da Posterior: 0.2611
## Mediana da Posterior: 0.2445
## Intervalo de Credibilidade 95%: [ 0.0852 , 0.5389 ]

```

Exercício 5

Num estudo para implementação de turbinas eólicas numa dada área mediu-se a velocidade do vento (em m/s) a uma dada altura ao longo de várias ocasiões e obteve os seguintes dados:

1.82, 1.09, 0.61, 0.04, 4.28, 1.03, 0.92, 0.99, 1.32, 3.20, 0.10, 0.57, 1.23, 0.26, 1.78.

O modelo que costuma ser utilizado para descrever esta variação é o modelo Weibull com parâmetros de escala e de forma denotados por δ e α , respectivamente, e cuja função densidade de probabilidade é expressa por

$$f(x|\delta, \alpha) = \delta \alpha x^{\alpha-1} e^{-\delta x^\alpha}, \quad x, \delta, \alpha > 0$$

Admita-se que a priori δ e α são independentes com distribuições Gama, $Ga(a, b)$, e LogNormal, $LN(c, d)$, de hiperparâmetros conhecidos, com $a, b, d > 0$ e $c \in \mathbb{R}$. Especifique os valores $a = 4$, $b = 1$, $c = -1$ e $d = 2$.

Supondo que os dados são uma concretização de uma amostra aleatória desse modelo:

a

Encontre as distribuições condicionais completas para os parâmetros δ e α .

Derivação 1: Condicional Completa de Delta

tratamos α como constante;

Da Verossimilhança (modelo Weibull):

$$\begin{aligned} L(\delta, \alpha|x) &= \prod_{i=1}^n \left[\delta \alpha x_i^{\alpha-1} e^{-\delta x_i^\alpha} \right] \\ L(\delta, \alpha|x) &= (\delta \alpha)^n \left(\prod_{i=1}^n x_i \right)^{\alpha-1} e^{-\delta \sum_{i=1}^n x_i^\alpha} \end{aligned}$$

Ignorando constantes em relação a δ :

$$L \propto (\delta)^n \cdot e^{-\delta \sum x_i^\alpha}$$

Da Priori de δ $Ga(a, b)$:

$$g(\delta) \propto \delta^{a-1} e^{-b\delta}$$

Portanto:

$$g(\delta|\alpha, x) \propto (\delta^n \cdot e^{-\delta \sum x_i^\alpha}) \times (\delta^{a-1} \cdot e^{-b\delta})$$

Simplificando:

$$g(\delta|\alpha, x) \propto \delta^{n+a-1} \cdot e^{-\delta(b+\sum x_i^\alpha)}$$

$$\text{logo } g(\delta|\alpha, x) \sim \text{Gamma}(n+a, b+\sum x_i^\alpha)$$

Usando os valores do problema ($n = 15$ dados, $a = 4$, $b = 1$):

$$\delta|\alpha, x \sim \text{Gama}(\text{shape} = 19, \text{rate} = 1 + \sum_{i=1}^{15} x_i^\alpha)$$

Derivação 2: Condicional Completa de Alfa

Da Verossimilhança (modelo Weibull):

$$\begin{aligned} L(\delta, \alpha|x) &= \prod_{i=1}^n \left[\delta \alpha x_i^{\alpha-1} e^{-\delta x_i^\alpha} \right] \\ L(\delta, \alpha|x) &= (\delta \alpha)^n \left(\prod_{i=1}^n x_i \right)^{\alpha-1} e^{-\delta \sum_{i=1}^n x_i^\alpha} \end{aligned}$$

Ignorando constantes em relação a α :

$$L \propto (\alpha)^n \cdot (\prod_{i=1}^n x_i)^{\alpha-1} \cdot e^{-\delta \sum x_i^\alpha}$$

Da Priori de α LN(c, d):

$$g(\alpha) \propto \frac{1}{\alpha} \exp\left(-\frac{(\ln(\alpha)-c)^2}{2d}\right)$$

Juntando temos:

$$g(\alpha|\delta, x) \propto \left[\alpha^n \cdot (\prod x_i)^{\alpha-1} \cdot e^{-\delta \sum x_i^\alpha} \right] \times \left[\frac{1}{\alpha} \cdot \exp\left(-\frac{(\ln(\alpha)-c)^2}{2d}\right) \right]$$

combinando α^n com $1/\alpha$:

$$g(\alpha|\delta, x) \propto \alpha^{n-1} \cdot (\prod x_i)^{\alpha-1} \cdot e^{-\delta \sum x_i^\alpha} \cdot \exp\left(-\frac{(\ln(\alpha)-c)^2}{2d}\right)$$

Portanto $g(\alpha|\delta, x)$ não segue distribuição conhecida.

Usando os valores do problema ($n = 15$, $c = -1$, $d = 2$):

$$g(\alpha|\delta, x) \propto \alpha^{14} \cdot \left(\prod_{i=1}^{15} x_i \right)^{\alpha-1} \cdot e^{-\delta \sum_{i=1}^{15} x_i^\alpha} \cdot \exp\left(-\frac{(\ln(\alpha)+1)^2}{4}\right)$$

b

Explique em detalhes como podemos gerar uma cadeia de Markov utilizando o amostrador de Gibbs para obtermos uma amostra da distribuição a posteriori conjunta de δ e α .

Com o que obtivemos no item ‘a’ temos que a condicional completa de δ é conhecida, mas a de α não. Portanto usaremos um Amostrador de Gibbs “modificado”;

1. Início

- Escolha um ponto inicial $\delta^{(0)}$ e $\alpha^{(0)}$. Esses valores devem ser > 0 ;
- Defina o N;
- Defina um tuning (usaremos para estimar α).

2. O Processo Iterativo

- Para $\delta(t)$
 - Pegamos o valor $\alpha^{(t-1)}$ da iteração anterior;
 - Calcule os parâmetros da Gama $g(\delta|\alpha^{(t-1)}, x) \sim \text{Gama}(\text{shape} = 19, \text{rate} = 1 + \sum_{i=1}^{15} x_i^{\alpha^{(t-1)}})$;
- Para $\alpha(t)$
 - Aqui usaremos Metropolis já que a distribuição não é cohecida;
 - Estado atual: $\alpha' = \alpha^{(t-1)}$;
 - A nossa “alvo” é a densidade condicional de α , chamaremos $h(\alpha)$:
$$h(\alpha) \propto \alpha^{14} \cdot (\prod x_i)^{\alpha-1} \cdot e^{-\delta^{(t)} \sum x_i^\alpha} \cdot \exp\left(-\frac{(\ln(\alpha)-c)^2}{4}\right);$$
 - Geramos um novo valor “candidato”, α_c . Com um random walk em torno do valor atual. Por exemplo, $\alpha_c \sim \text{Normal}(\text{média} = \alpha', \text{sd} = \sigma_{\text{prop}})$;
 - Calcule a Razão de Aceitação $R = \frac{h(\alpha_c)}{h(\alpha_{\text{atual}})}$;
 - Geramos um número aleatório u de uma Uniforme(0, 1);
 - Aceitamos se $u < \min(1, R)$: $\therefore \alpha^{(t)} = \alpha_c$, C.C. rejeitamos $\alpha^{(t)} = \alpha^{(t-1)}$.

c

Construa um código comentado para gerar uma amostra da distribuição a posteriori conjunta de δ e α a partir do algoritmo obtido no item anterior.

```
set.seed(52)
# --- Dados e Hiperparâmetros ---

x <- c(1.82, 1.09, 0.61, 0.04, 4.28, 1.03, 0.92, 0.99, 1.32, 3.20, 0.10, 0.57, 1.23, 0.26, 1.78)
n <- length(x)

# Hiperparâmetros das priori
# delta ~ Gama(a, b)
a <- 4
b <- 1
# alpha ~ LogNormal(c, d)
d <- 2
c <- -1

# Pré-calcular a soma dos logs de x (usado na densidade de alpha)
sum_log_x <- sum(log(x))

# Função da Log-Densidade a Posteriori (Kernel) para Alpha
# Esta é a função g(alpha / delta, x) que encontramos no item (a),
# mas em escala logarítmica
```

```

#
#  $\log(g(\alpha / \dots)) \backslash prop (n-1)*\log(\alpha) + (\alpha-1)*\sum(\log(x)) -$ 
#  $\delta*\sum(x^{\alpha}) - (\log(\alpha)-c)^2 / (2*\delta)$ 
#
log_post_alpha <- function(alpha, delta, x_dados, n_obs, c_prior, d_prior, s_log_x) {

  # Alpha deve ser positivo (domínio da Weibull e LogNormal)
  if (alpha <= 0) {
    return(-Inf) # Retorna log(0), que é -Infinito
  }

  # Calcular a soma de  $x^{\alpha}$ 
  sum_x_alpha <- sum(x_dados^alpha)

  # Termos da log-verossimilhança proporcional
  log_lik_1 <- (n_obs - 1) * log(alpha)
  log_lik_2 <- (alpha - 1) * s_log_x
  log_lik_3 <- -delta * sum_x_alpha

  # Termo da log-priori proporcional
  log_prior <- -(log(alpha) - c_prior)^2 / (2 * d_prior)

  # A soma de todos os termos é a log-densidade a posteriori
  return(log_lik_1 + log_lik_2 + log_lik_3 + log_prior)
}

# Configuração do Amostrador MCMC ---
# Número total de iterações
N_total <- 15000

# Vetores para armazenar as amostras (cadeias)
# Começamos com NA e preenchemos o primeiro valor
delta_chain <- rep(NA, N_total)
alpha_chain <- rep(NA, N_total)

# Valores iniciais (chutes)
delta_chain[1] <- 1.0
alpha_chain[1] <- 1.0

# Parâmetro de ajuste (tuning) para a proposta do Metropolis-Hastings
# Esta é a variância da proposta Normal para alpha.
# Ajustar isso é crucial para a eficiência do amostrador.
# (Este valor é encontrado por tentativa e erro)
sigma_prop_alpha <- 0.15

# Variável para rastrear a taxa de aceitação (bom para diagnóstico)
aceitos <- 0

# O Loop do Amostrador (Metropolis-within-Gibbs) ---

```

```

print("Iniciando a amostragem MCMC...")

for (t in 2:N_total) {

  # Amostrar Delta (Passo Gibbs) ---

  # valor anterior de alpha
  alpha_anterior <- alpha_chain[t-1]

  # Calcule os parâmetros da condicional Gama de delta
  # shape = n + a
  shape_gama <- n + a # 15 + 4 = 19
  # rate = b + sum(x^alpha_anterior)
  rate_gama <- b + sum(x^alpha_anterior)

  # Amostre delta e armazene
  delta_atual <- rgamma(1, shape = shape_gama, rate = rate_gama)
  delta_chain[t] <- delta_atual

  # Amostrar Alpha (Passo Metropolis-Hastings) ---

  # Pegue o valor anterior de alpha
  alpha_anterior <- alpha_chain[t-1]

  # Proponha um novo alpha (candidato)
  # Usamos um "passeio aleatório" (random walk) Normal
  alpha_candidato <- rnorm(1, mean = alpha_anterior, sd = sigma_prop_alpha)

  # Calcule a razão de aceitação (em log-escala)

  # Se o candidato for inválido (negativo), a probabilidade é 0 (log é -Inf)
  # e a razão de aceitação será 0, então rejeitamos automaticamente.
  if (alpha_candidato <= 0) {
    log_R <- -Inf
  } else {
    # Calcule a log-densidade no ponto candidato
    log_H_candidato <- log_post_alpha(
      alpha = alpha_candidato,
      delta = delta_atual,
      x_dados = x, n_obs = n, c_prior = c, d_prior = d, s_log_x = sum_log_x
    )

    # Calcule a log-densidade no ponto anterior
    log_H_anterior <- log_post_alpha(
      alpha = alpha_anterior,
      delta = delta_atual,
      x_dados = x, n_obs = n, c_prior = c, d_prior = d, s_log_x = sum_log_x
    )

    # Razão de Aceitação (log_R = log(H_c) - log(H_a))
    # A razão da proposta q(a/c) / q(c/a) é 1
    log_R <- log_H_candidato - log_H_anterior
  }
}

```

```

# 4. Decida (Aceitar ou Rejeitar)
# Sorteia um número  $u \sim \text{Uniforme}(0, 1)$ 
# Aceita se  $\log(u) < \log_R$  (que é o mesmo que  $u < R$ )

if (log(runif(1)) < log_R) {
  # Aceita o candidato
  alpha_chain[t] <- alpha_candidato
  aceitos <- aceitos + 1
} else {
  # Rejeita o candidato (mantém o valor anterior)
  alpha_chain[t] <- alpha_anterior
}
}

print("Amostragem concluída.")

# Resultado ---
print(paste("Taxa de Aceitação para Alpha:", round(aceitos / (N_total - 1), 3)))
print("Últimos 5 valores da cadeia de Delta:")
print(tail(delta_chain, 5))
print("Últimos 5 valores da cadeia de Alpha:")
print(tail(alpha_chain, 5))

## [1] "Iniciando a amostragem MCMC..."
## [1] "Amostragem concluída."
## [1] "Taxa de Aceitação para Alpha: 0.753"
## [1] "Últimos 5 valores da cadeia de Delta:"
## [1] 0.9874039 0.8307836 1.0515809 0.5601584 0.9989536
## [1] "Últimos 5 valores da cadeia de Alpha:"
## [1] 1.0894724 1.0343642 1.1157348 0.9073305 1.0308984

```

d

Gere uma trajetória da cadeia a partir do código e dos dados, especificando o tamanho da cadeia, o período de burn-in, o thinning (lag) e os valores iniciais.

```

# (Este código é a continuação do item 'c')

# Análise de Convergência e Inferência (Item d)
# install.packages("coda")
library(coda)

# Definir Parâmetros de Análise ---

# Total de amostras geradas
N_total <- 15000
# Período de burn-in
burn_in <- 5000
# Thinning (lag)
thinning <- 10

# Criar Objetos MCMC (pós-processamento) ---

# Primeiro, aplicamos o burn-in (descartamos os primeiros 'burn_in' valores)

```

```

delta_post_burnin <- delta_chain[(burn_in + 1):N_total]
alpha_post_burnin <- alpha_chain[(burn_in + 1):N_total]

# Em seguida, aplicamos o thinning (pegamos 1 a cada 'thinning' valores)
indices <- seq(1, length(delta_post_burnin), by = thinning)

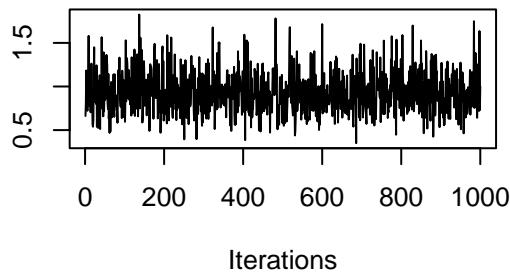
delta_final_chain <- delta_post_burnin[indices]
alpha_final_chain <- alpha_post_burnin[indices]

# (Combinamos as cadeias em uma matriz)
combined_chains <- cbind(delta = delta_final_chain, alpha = alpha_final_chain)
mcmc_object <- as.mcmc(combined_chains)

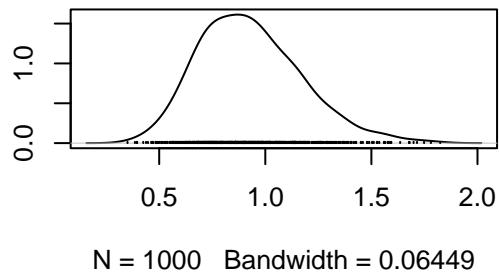
# Avaliação da Convergência ---
# INTERPRETAÇÃO:
# 1. Trace Plots:
#     sem tendências de subida ou descida. Isso indica que a cadeia
#     está estável.
# 2. Densidade: Deve ser suave, indicando que a cadeia
#     explorou bem a distribuição.
plot(mcmc_object)

```

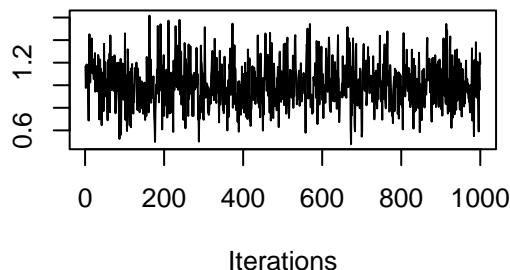
Trace of delta



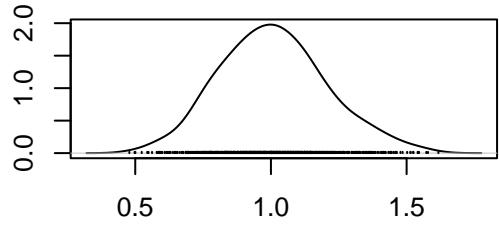
Density of delta



Trace of alpha



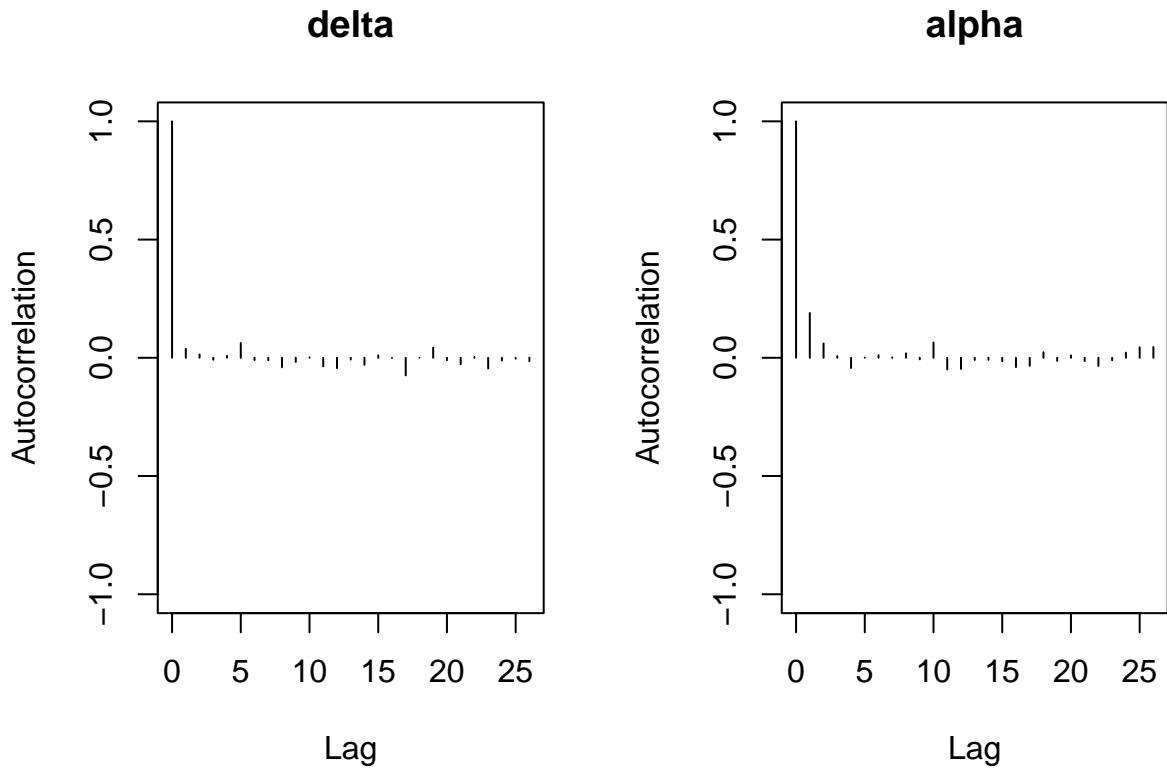
Density of alpha



```

# Queremos que a correlação caia para 0 rapidamente.
autocorr.plot(mcmc_object)

```



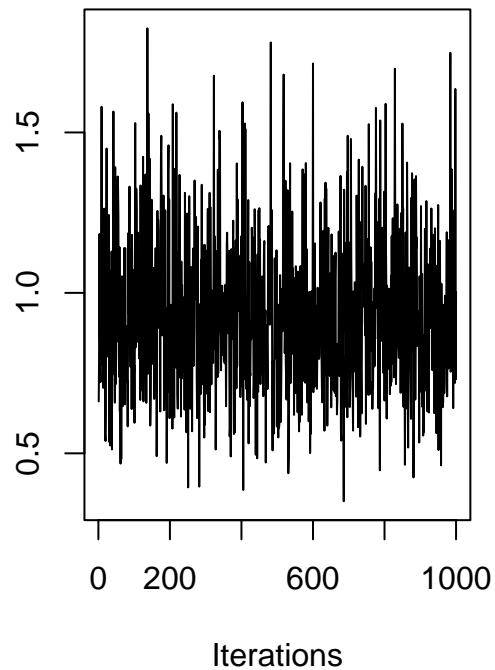
e

Avalie a convergência da cadeia gerada e realize a inferência para os parâmetros; utilize estimativas pontuais, intervalos de credibilidade e construa gráficos.

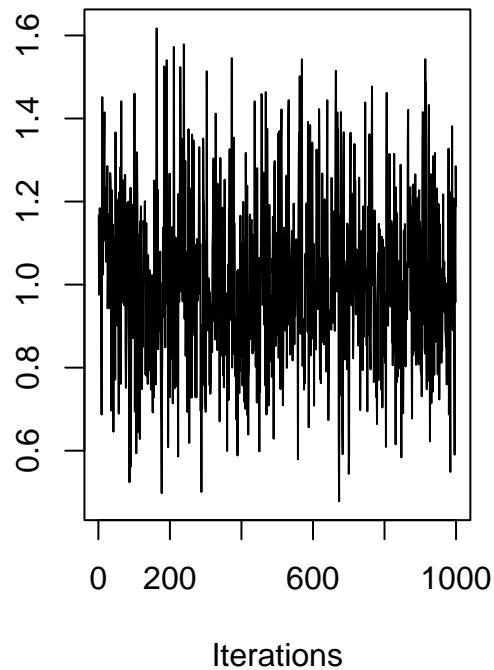
Conforme os gráficos do item anterior:

```
plot(mcmc_object, density = FALSE)
```

Trace of delta



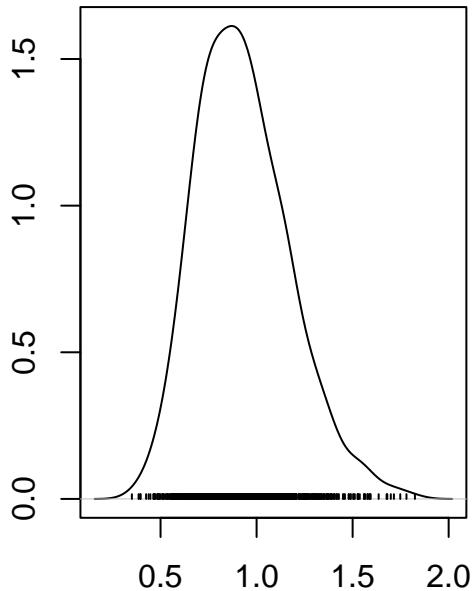
Trace of alpha



Trace plots não apresentam tendências de subida/descida, portanto a cadeia convergiu.

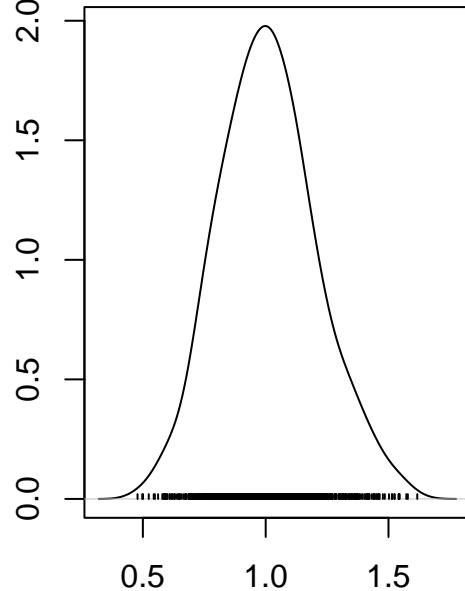
```
plot(mcmc_object, trace = FALSE)
```

Density of delta



N = 1000 Bandwidth = 0.06449

Density of alpha



N = 1000 Bandwidth = 0.05267

Densidades formam uma curva suave sem picos irregulares, portanto a distribuição foi bem explorada.

Inferência:

```

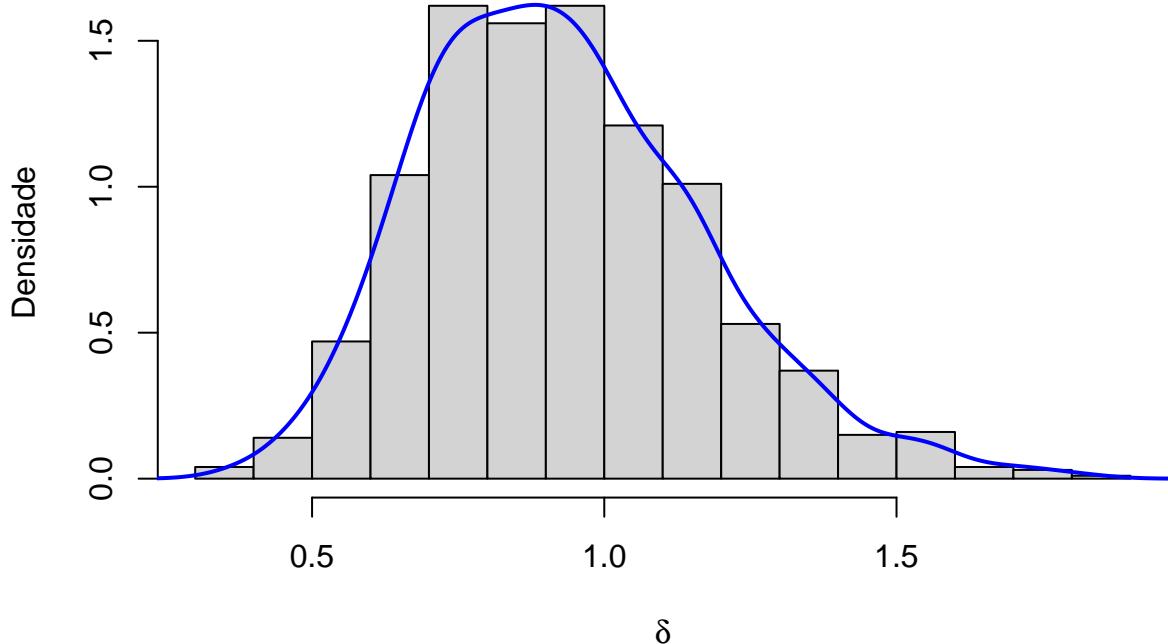
# Gráficos das Posteriori (Inferência Visual)
# (Já foram criados pelo 'plot(mcmc_object)', mas podemos fazê-los melhor)

#par(mfrow = c(1, 2))

# Histograma e Densidade para Delta
hist(delta_final_chain, freq = FALSE, main = "Posteriori de Delta",
      xlab = expression(delta), ylab = "Densidade")
lines(density(delta_final_chain), col = "blue", lwd = 2)

```

Posteriori de Delta

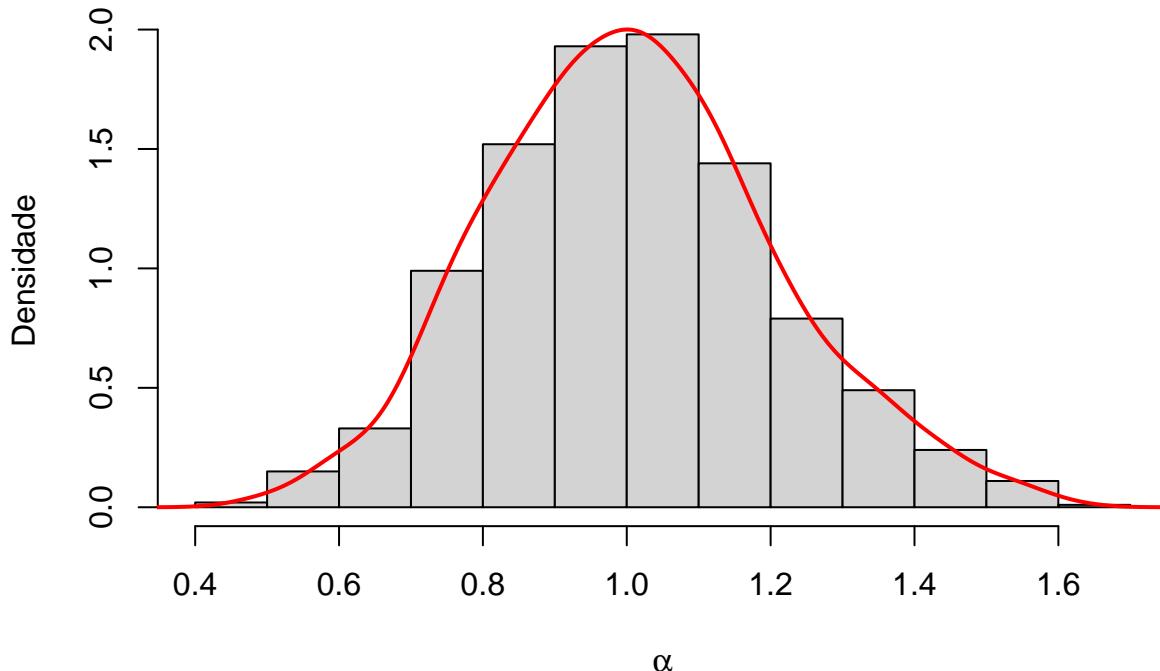


```

# Histograma e Densidade para Alpha
hist(alpha_final_chain, freq = FALSE, main = "Posteriori de Alpha",
      xlab = expression(alpha), ylab = "Densidade")
lines(density(alpha_final_chain), col = "red", lwd = 2)

```

Posteriori de Alpha



```
#par(mfrow = c(1, 1))

# Estimativas Pontuais e Intervalos de Credibilidade
# - Média (Mean): Estimativa pontual comum
# - Mediana (Median): Outra estimativa pontual, robusta a assimetrias
# - Quantis (2.5% e 97.5%): Este é o nosso Intervalo de Credibilidade de 95%
summary(mcmc_object)

##
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD Naive SE Time-series SE
## delta  0.9291  0.2422  0.007659      0.007659
## alpha   1.0086  0.1984  0.006275      0.007604
##
## 2. Quantiles for each variable:
##
##        2.5%    25%    50%    75% 97.5%
## delta  0.5309  0.7506  0.9047  1.079  1.490
## alpha   0.6385  0.8682  1.0021  1.133  1.433
```

Temos 95% de probabilidade de que o verdadeiro valor do parâmetro α esteja contido no intervalo [0.63 , 1.43].

Temos 95% de probabilidade de que o verdadeiro valor do parâmetro δ esteja contido no intervalo [0.53 , 1.49].